Date: 29/07/2024

WEEK 3: Hyperparameter tuning, Batch Normalization, Programming frameworks

Hyperparameters:

- α → most important
- β → 0.9 most default
- β_1 (0.9), β_2 (0.999), ε (10^{-8})
- # layers
- # hidden units
- learning rate decay
- mini-batch size

Priority:

1.
2.
3.

Try random values: Don't use a grid (fixed values)

Date:

Fig 1.

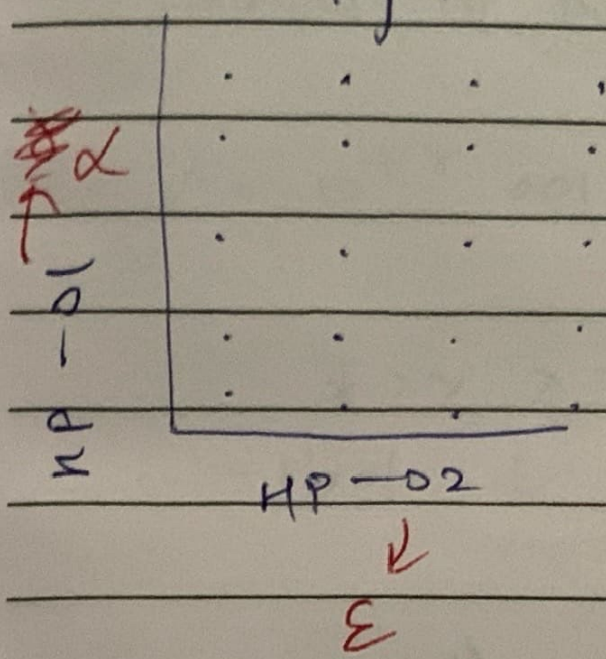
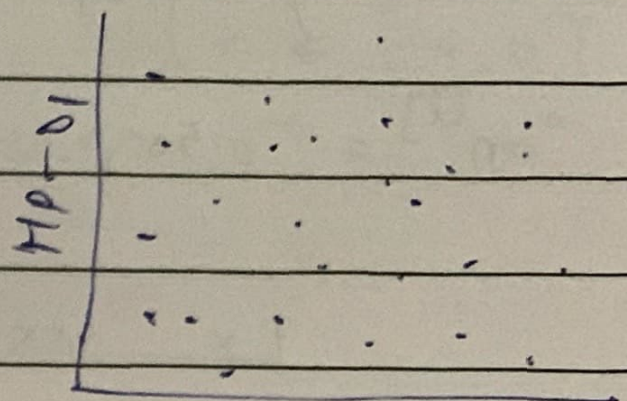


Fig 2.



In Fig 1., the values of alpha can be tried is only 5, so α and ε get equal importance even when importantance of α >>>> ε.

⇒ Coarse to fine:

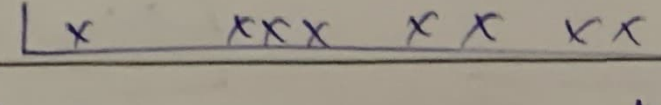
We zoom in into a region where the points tend to work well. In this chosen region we find more such points similarly for best accuracy and pair.

∴ We can sample more densely now.

Date:

→ Picking hyperparameters at random:

$n^{[l]} = 50, \ldots, 100.$

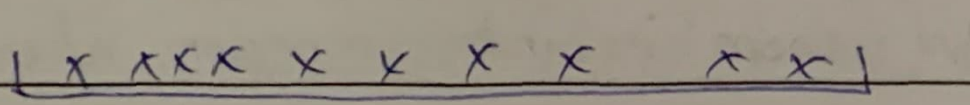


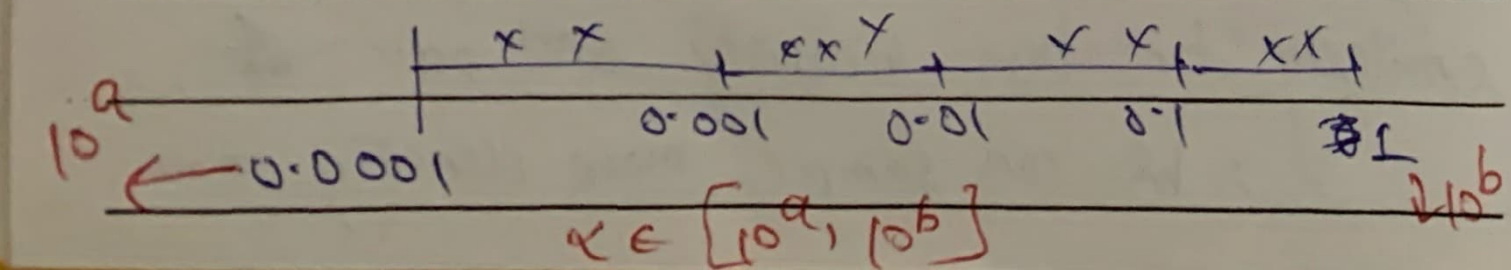
layers L : 2 – 4

Uniformly random is fine for these no. of layers.

→ Apt. scale for hyperparameters:

$\alpha = 0.0001, \ldots, 1$





Date:

$r = -4 * np.random.rand()$

$r \in [-4, 0]$

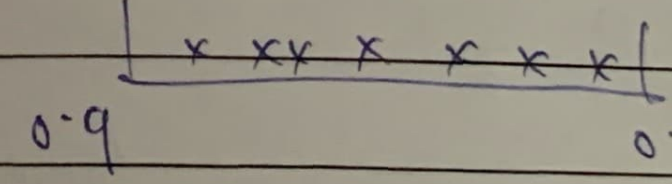
$\alpha = 10^{+r}$ # $10^{-4} \dots 10^{0}$

→ HiPs for exponentially weighted averages

$\beta = 0.9 \dots 0.999$

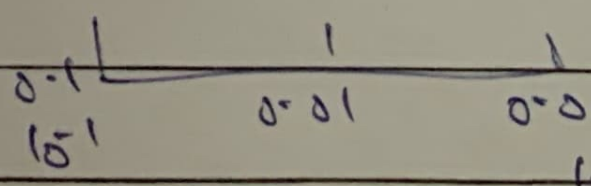
↓ avg. over last 10 units

↓ avg. over last 1000 units



← β

$= 0.1 \dots 0.001$



Date:

$\therefore r \in [-3, -1]$

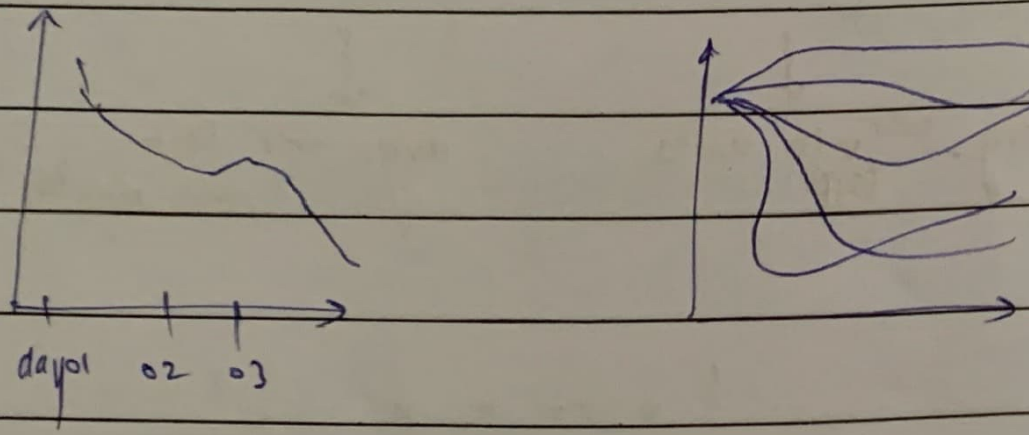
$1 - \beta = 10^r$

$\beta = 1 - 10^r$

small change in beta close to 1 is very sensitive, this helps in ~~:~~ samples more densely when $\beta \approx 1$

Baby sitting one model

Training many models in parallel



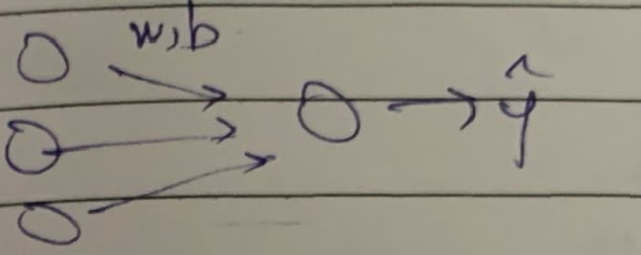
Panda

Caviar

If CPU's GPU's are not limited, use Caviar approach.

Date:

⇒ Normalising inputs to speed up learning

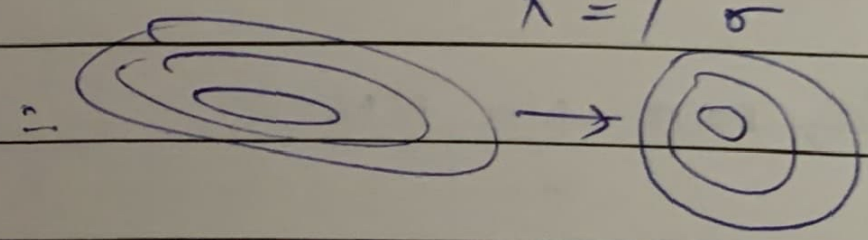


$$\mu = \frac{1}{m} \sum_{i=1} x^{(i)}$$

$$x = x - \mu$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1} (x^{(i)})^2 \quad \text{element-wise}$$

$$x = x / \sigma$$



⇒ Implementing a Batch norm :

Given some intermediate values in NN $z^{(1)}, \ldots, z^{(m)}$

$\rightarrow z^{[l](i)}$

$$\mu = \frac{1}{m} \sum_{i=1} z^{(i)}$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1} \left(z^{(i)} - \mu\right)^2$$

$$z^{(i)}_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

Here mean becomes $\rightarrow 0$, variance $\rightarrow 1$ for all layers but we don't want this

learnable parameters of model

Date:

$$\therefore \tilde{Z}^{(i)} = \gamma Z^{(i)}_{norm} + \beta$$

this helps in setting variance and mean to our choice.

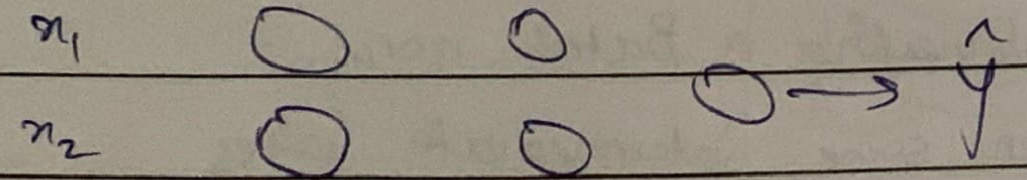
if $\gamma = \sqrt{\sigma^2 + \varepsilon}$

and $\beta = \mu$

then

$$\tilde{Z}^{(i)} = Z^{(i)}$$

→ Adding Batch norm to a NN:



$$X \xrightarrow{w^{[1]}, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm (BN)}]{\beta^{[1]}, \gamma^{[1]}} \tilde{Z}^{[1]} \rightarrow a^{[1]} = g^{[1]}(\tilde{Z}^{[1]})$$

$$a^{[1]} \xrightarrow{w^{[2]}, b^{[2]}} Z^{[2]} \xrightarrow[BN]{\beta^{[2]}, \gamma^{[2]}} \tilde{Z}^{[2]} \rightarrow a^{[2]} \rightarrow \cdots$$

Date:

Parameters : $w^{[1]}, b^{[1]} \ldots, w^{[L]}, b^{[L]}$

$\beta^{[1]}, \gamma^{[1]} \ldots \beta^{[L]}, \gamma^{[L]}$

→ working with mini-batches

$X^{\{1\}} \xrightarrow{w^{[1]}, b^{[1]}} z^{[1]} \xrightarrow[BN]{\beta^{[1]}, \gamma^{[1]}} \tilde{z}^{[1]} \rightarrow a^{[1]} = g^{[1]}(\tilde{z}^{[1]})$

$a^{[1]} \xrightarrow{w^{[2]}, b^{[2]}} z^{[2]} \rightarrow \ldots$

$X^{\{2\}} \rightarrow z^{[1]} \xrightarrow[BN]{\beta^{[1]}, \gamma^{[1]}} \tilde{z}^{[1]} \rightarrow \ldots$

⋮

Parameters : $w^{[l]}, b^{[l]}, \beta^{[l]}, \gamma^{[l]}$ → $(n^{[l]}, 1)$

during BN, $b^{[l]}$ can be eliminated (or be set permanently to zero.

because batch-norm zeroes out the mean

∴ No point in adding everywhere : if is going to be

Date:

Implementing grad. descent:

for $t = 1 \ldots$ num MiniBatches

Compute forward prop on $X^{\{t\}}$

In each hidden layer, use BN

to replace $z^{[l]}$ with $\tilde{z}^{[l]}$

Use back prop to compute

$dw^{[l]}, d\beta^{[l]}, d\gamma^{[l]}$

Update parameters

$$w^{[l]} := w^{[l]} - \alpha\, dw^{[l]}$$

$$\beta^{[l]} := \beta^{[l]} - \alpha\, d\beta^{[l]}$$

$$\gamma^{[l]} := \gamma^{[l]} - \alpha\, d\gamma^{[l]}$$

Works with momentum, RMS prop, Adam.

Date:

→ When the distribution of the input set changes, the parameters learnt further change, and ∴ the deep NN suffers a covariate shift, causing less accuracy and the cost function might not minimise well.

∴ If batch norm is applied, the distributions are normalised helping the model to learn the parameters in a better way.

→ Batch norm as regularization:

(1) Each mini-batch is scaled by mean/variance computed on just that mini-batch

(2) This adds some noise to the values $z^{[l]}$ within that mini-batch. So similar to dropout, it adds some noise to each hidden layer's activation.

(3) This has a slight regularization effect

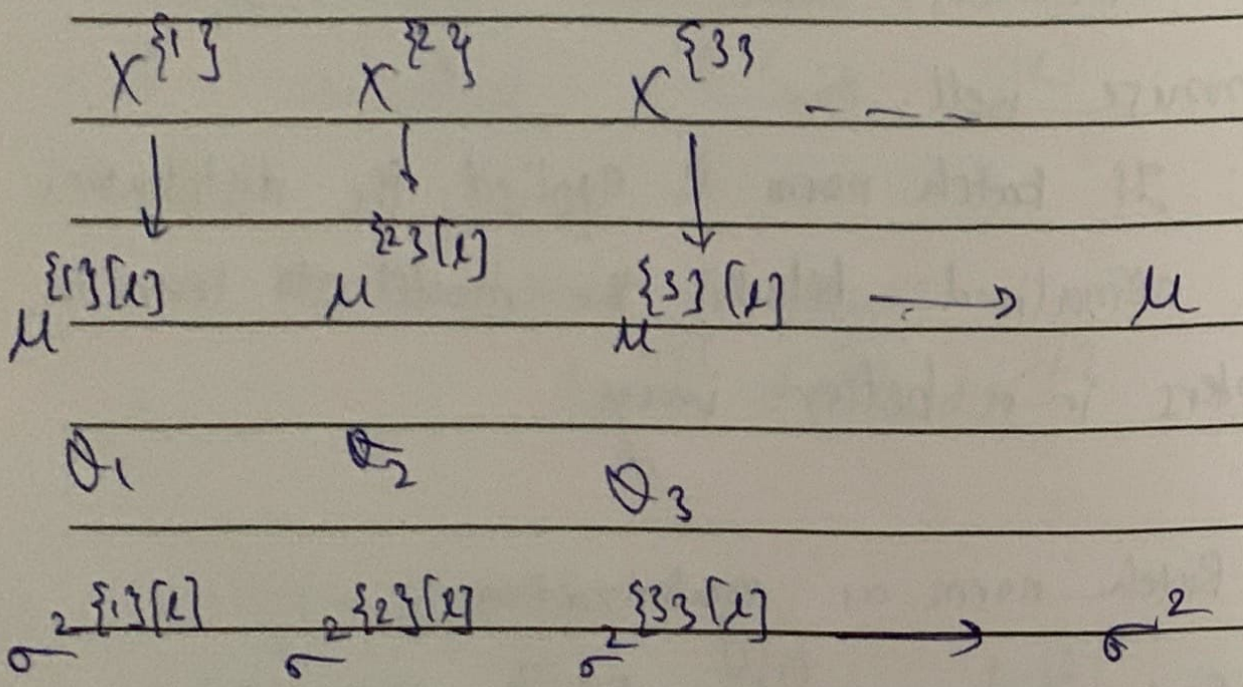
batch-norm + dropout
= powerful regularization

Date:

→ Batch-norm at test time:

(Running average)

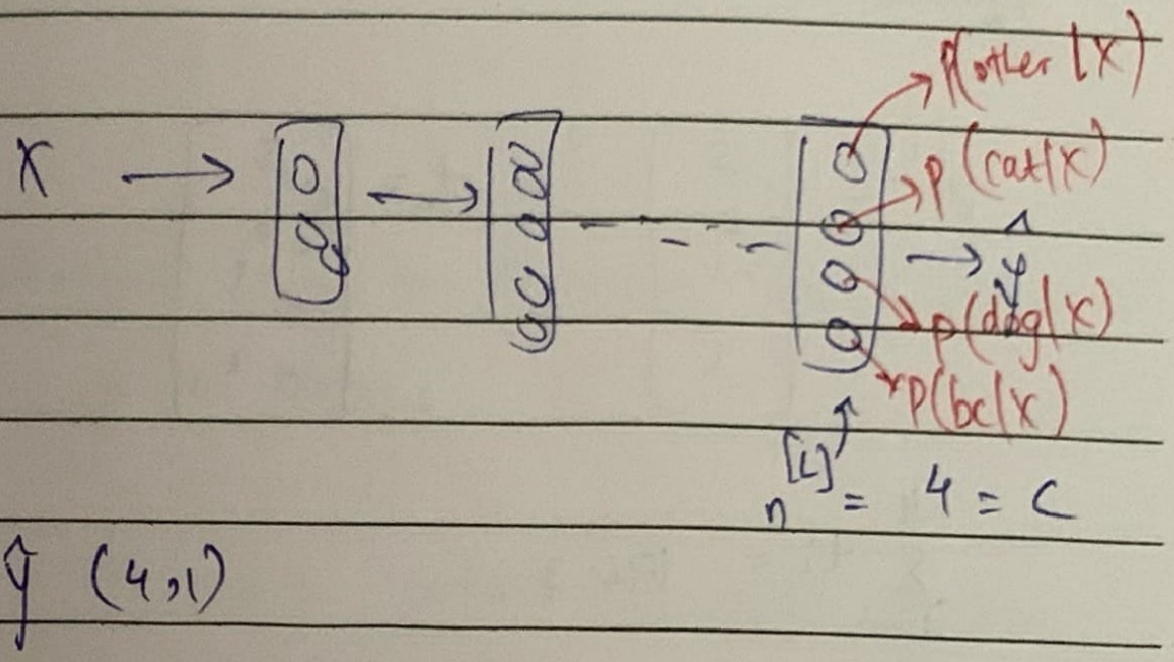
μ, σ^2: estimate using exponentially weighted average (across mini-batches)



Date:

→ Softmax regression: (generalization of LR with ~~two~~ or more classes)

C = # classes = 4 (0, ..., 3)



Softmax layer:

$$z^{[L]} = w^{[L]} a^{[L-1]} + b^{[L]} \quad (4,1)$$

Activation function:

$$t = e^{z^{[L]}} \quad (4,1)$$

Date:

$$a^{[L]} = \frac{e^{z^{[L]}}}{\sum_{i=1}^{4} t_i} \qquad a_i^{[L]} = \frac{t_i}{\sum_{i=1}^{4} t_i}$$

Example:

$$z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix} \qquad t = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix}$$

$$\sum_{i=1}^{4} t_i = 176.3$$

$$a^{[L]} = \frac{t}{176.3} \rightarrow \text{as a vector}$$

individually;

for $i=1$ $\quad e^5/176.3$

$=2$ $\quad e^2/176.3$

$=3$ $\quad e^{-1}/176.3$

$=4$ $\quad e^3/176.3$

} These sum to 1.

SOFTMAX:

Date: 30/07/2024

Softmax examples: Shown in videos.

"hard max" $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ → given to the highest value

Softmax regression generalises logistic regression to C classes.

If C = 2, softmax = logistic regression

Loss function:

$$y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad a^{[L]} = \hat{y} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0.4 \end{bmatrix}$$

C=4 here Date:

$$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

$$= -y_2 (\log \hat{y}_2) = -\log \hat{y}_2$$

To make $-\log \hat{y}_2 \downarrow$

make $\hat{y}_2 \uparrow$

$$J(w^{[1]}, b^{[1]} \dots) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

$$Y = [y^{(1)} \; y^{(2)} \dots y^{(m)}]$$

$$= (C, m)$$

$$\hat{Y} = [\hat{y}^{(1)} \; \hat{y}^{(2)} \dots \hat{y}^{(m)}]$$

$$= (C, m)$$

Date:

Softmax layer outputs

$$z^{[L]} \rightarrow a^{[L]} = \hat{y} \rightarrow \mathcal{L}(\hat{y}, y)$$

Back prop:

$$dz^{[L]} = \hat{y} - y$$